## CLAIMS

1. (Original) A resource management and task allocation controller for a multicore processor having a plurality of interconnected processor elements, at least one of which is a master processing unit, each element providing resources for processing executable transactions, the controller being in communication with each of the processor elements but separate from the master processing unit, and comprising control logic to allocate executable transactions within the multicore processor to a one of the processor elements in accordance with one of a range of pre-defined allocation parameters.

2. (Original) The controller of claim 1, wherein the range of predefined allocation parameters included within the control logic of the controller contains a plurality of transaction scheduling rules, for scheduling the timing and/or order of execution of the executable transactions by the processor elements.

3. (Original) The controller of claim 1, wherein the range of predefined allocation parameters included within the control logic of the controller contains a plurality of system management rules, for controlling the manner in which the executable transactions are executed by the processor elements.

4. (Original) The controller of claim 2, further configured to generate instructions for communication to the processing elements.

5. (Original) The controller of claim 4, configured to send a processor element configuration instruction to a processor element, which causes the said processor element in turn to be adapted so as to permit subsequent execution of an executable transaction allocated to that processor element by the controller.

6. (Original) The controller of claim 4, configured to generate instructions by the transmission of one or more interrupts to the processor elements.

7. (Original) The controller of claim 1, wherein said control logic further comprises:

an executable transaction manager; and;

a dedicated memory manager;

wherein the said dedicated memory manager controls access by the executable transaction manager to the dedicated memory.

8. (Original) The controller of claim 7, wherein the executable transaction manager further comprises an executable transaction input manager,

configured to maintain an indication of available memory within the dedicated memory.

9. (Original) The controller of claim 8, wherein the executable transaction manager input is configured to maintain a list of available memory locations within the dedicated memory.

10. (Original) The controller of claim 9, wherein the executable transaction input manager maintains the indication of available memory as a result of updated instructions from the dedicated memory manager.

11. (Original) The controller of claim 1, wherein the executable transaction to be allocated include threads, each of which form part of an application being executed upon the multicore processor, wherein at least some of the threads are independent threads capable of execution independently of other events, and wherein at least some of the threads are dependent threads, whose execution is dependent upon the existence of a predetermined event.

12. (Original) The controller of claim 11, wherein the control logic further comprises a time manager configured to provide timer functions to said executable transaction manager.

13. (Original) The controller of claim 12, wherein the predetermined event is a timing event.

14. (Original) The controller of claim 11, wherein the predetermined event is the completion of the execution of a previous thread.

15. (Original) The controller of claim 11, wherein the executable transaction manager further comprises an executable transaction synchronisation manager, configured to maintain at least one pending queue list within the dedicated memory, indicative of dependent threads awaiting the occurrence of a predetermined event, and at least one timer queue list within the dedicated memory, indicative of threads awaiting a timing event.

16. (Original) The controller of claim 15, wherein the executable transaction manager further comprises an executable transaction output manager configured to maintain a plurality of dispatch queue structures within the dedicated memory, indicative of the threads awaiting execution on an associated one of the processor elements, and to maintain a plurality of ready queue structures within the dedicated memory, indicative of threads awaiting allocation to a one of the processor elements for execution there.

17. (Original) The controller of claim 16, wherein the executable transaction manager further comprises an executable transaction schedule manager, configured to provide and maintain scheduling decisions for prioritising the dispatch of threads from within the ready queues to the dispatch queue for each processor element.

18. (Original) The controller of claim 1, wherein the control logic further comprises a system interface manager, in communication with the executable transaction manager, and configured to manage access by the controller to the multicore processor.

19. (Original) The controller of claim 18, wherein the system interface manager is arranged to provide interconnect interfacing and configuration and run-time access to said executable transaction manager.

20. (Original) The controller of claim 6, wherein the control logic further comprises a system interrupt manager, for converting system interrupts in a first format employed within the multicore processor, into controller interrupts in a second, different format, which second format is understandable by the executable transaction manager.

21. (Previously Presented) A multicore processor comprising:

a plurality of interconnected processor elements, at least one of which is a master processing unit, each element providing resources for processing executable transactions;

a resource management and task allocation controller, in communication with each of the processor elements but separate from the master processing unit, and comprising control logic for allocating executable transactions within the multicore processor to a one of the processor elements in accordance with one of a range of pre-defined allocation parameters; and

a plurality of controller clients, at least one of which is associated with a corresponding one of the plurality of interconnected processor elements, wherein each controller client is configured to control communications between its said associated processing element and the rest of the multicore processor, dependent upon control signals from the task allocation controller.

22. (Original) The multicore processor of claim 21, further comprising a shared system bus accessible by both the controller and the plurality of interconnected processor elements.

23. (Original) The processor of claim 22, further comprising an external interface, for connecting said multicore processor to one or more external devices:

24. (Previously Presented) The multicore processor as claimed in claim 21, further comprising a dedicated memory in communication with the controller.

25. (Original) The multicore processor of claim 24, wherein the dedicated memory is exclusively accessible by the controller.

26. (Previously Presented) The multicore processor of claim 24, wherein the dedicated memory is accessible by both the controller and by at least one further component of the multicore processor.

27. (Previously Presented) The multicore processor of claim 24, wherein the dedicated memory comprises a plurality of individual memory elements.

28. (Original) The multicore processor of claim 27, wherein the number of individual memory elements is user definable.

29. (Original) The multicore processor of claim 28, wherein each memory element is of a similar size and the user definable number of memory elements results in a variable memory size.

30. (Original) The multicore processor of claim 21, wherein the, or at least one of the controller clients is a software application running on the associated processor element.

31. (Original) The multicore processor of claim 21, wherein the, or at least one of the controller clients is a hardware controller client, dependent on the functionality of the associated processor element.

32. (Original) The multicore processor of claim 31, wherein each controller client further comprises a client control logic, for controlling the associated processor element, upon activation by a control signal from the said controller.

33. (Original) The multicore processor of claim 32, wherein each controller client further comprises a plurality of buffers, for temporary storage of communication messages sent between the said processor element and the rest of the multicore processor.

34. (Original) The multicore processor of claim 33, wherein the plurality of buffers are first in first out buffers.

35. (Original) The multicore processor of claim 33, wherein each controller client further comprises a plurality of memory elements, each configured to store an address.

36. (Original) The multicore processor of claim 35, wherein each controller client further comprises a plurality of comparators, each comparator configured to compare an address generated by the associated processor element with an address stored in a one of the memory elements.

37. (Original) The multicore processor of claim 35, wherein each controller client further comprises a subtractor, configured to subtract an address stored in a one of the memory elements from an address generated by the associated processor element.

38. (Original) A method of controlling and allocating resources within a multicore processor having a plurality of processor elements, at least one of which is a master processing unit, comprising:
receiving an executable transaction at a resource management and task allocation controller separate from the master processor unit; and
allocating that executable transaction to a one of the processor elements independently of central processing unit control.

39. (Original) The method of claim 38, further comprising:

directing the executable transaction to a one of the processor elements via a transaction management client.

40. (Original) The method of claim 39, wherein the transaction management client is a hardware client.

41. (Original) The method of claim 39, wherein the transaction management client is a software client.

42. (Original) The method of claim 40, further comprising storing a predetermined address within the transaction management client.

43. (Original) The method of claim 42, further comprising: at the transaction management client, subtracting the predetermined address from an address generated by the associated processing element to produce a normalised address.

44. (Original) The method of claim 42, further comprising, at the transaction management client, comparing an address generated by the associated processor element with the stored predetermined address; and configuring the processor element dependent on the result of the comparison.

45. (Original) The method of claim 40, further comprising, at the transaction management client, storing the whole of a communication message from the rest of the multicore processor to the associated processor element; and
subsequently passing the whole message to the associated processor element.

46. (Original) The method of claim 40, further comprising, at the transaction management client, streaming communication messages from the rest of the multicore processor to the associated processor element.

47. (Original) The method of claim 39, further comprising creating, executing or deleting an executable transaction for a first transaction management client, with a second transaction management client.

48. (Original) The method of claim 38, further comprising allocating the executable transaction to a one of the processing elements based upon a pre-defined set of scheduling parameters.

49. (Original) The method of claim 48, wherein the set of scheduling parameters is user-definable.

50. (Original) The method of claim 48, further comprising:

monitoring a list of the scheduling parameters for use by the controller

51. (Original) The method of claim 48, further comprising altering the set of scheduling parameters over time.

52. (Original) The method of claim 50, wherein the step of maintaining the list of the scheduling parameters further comprises maintaining a list of ready tasks that may be carried out by one or more of the processor elements.

53. (Original) The method of claim 48, further comprising allocating the executable transaction to a one of the processing elements on the basis of the requirement to balance processor resources within the multicore processor.

54. (Original) The method of claim 48, further comprising preventing the allocation of the executable transaction to a one of the processor elements, when it is determined that it is desirable for that processor element to execute a higher priority task.

55. (Original) The method of claim 38, further comprising:

maintaining a list of executable transactions that have not been allocated for longer that a predetermined length of time.

56. (Original) The method of claim 50, wherein the step of monitoring the Fist of the scheduling parameters further comprises maintaining a list of pending tasks that are awaiting a predetermined event.

57. (Original) The method of claim 56, wherein the predetermined event is a timer event, a synchronisation event or both.

58. (Original) The method of claim 56, further comprising maintaining a plurality of lists of pending tasks, according to mutually exclusive predetermined events.

59. (Original) The method of claim 50, wherein the step of monitoring the list of the scheduling parameters further comprises maintaining a list of dispatched tasks that are awaiting execution on a particular processing resource.

60. (Original) The method of claim 59, further comprising the step of moving a executable transaction awaiting a predetermined event to the dispatch queue, on expiration of a timeout.

61. (New) The method of claim 21, wherein said resource management and task allocation controller comprises an external interrupt control logic for processing external interrupts through a priority structure, wherein said external control logic controllably interrupts said plurality of processing elements based on said priority structure.